DOCUMENT RESUME

ED 479 604 IR 022 208

AUTHOR	Rochefort, Stephen
TITLE	An Agent Architecture for Asynchronous Learning.
PUB DATE	2002-10-00
NOTE	8p.; In: E-Learn 2002 World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education. Proceedings (7th, Montreal, Quebec, Canada, October 15-19, 2002); see IR 022 093.
AVAILABLE FROM	Association for the Advancement of Computing in Education (AACE), P.O. Box 3728, Norfolk, VA 23514. Tel: 757-623-7588; e-mail: info@aace.org; Web site: http://www.aace.org/DL/.
PUB TYPE	Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE	EDRS Price MF01 Plus Postage. PC Not Available from EDRS
DESCRIPTORS	*Artificial Intelligence; Computer Assisted Instruction; Computer Interfaces; Computer Mediated Communication; *Computer System Design; Cooperative Learning; *Distance Education; Educational Environment; Educational Technology; Instructional Materials

ABSTRACT

Education delivery systems have evolved from content delivery mechanisms to collaborative environments. The next step in this evolution is to provide greater support for asynchronous learners, those that progress through course material with little or no support from the instructional team. The integration of Web technologies with artificial intelligence techniques and multi-agent environments provides an architecture that supports the development of asynchronous-based systems. Autonomous agents are capable of providing ease of use and guidance through the introduction of natural language interfaces, knowledge processing, inferential reasoning, and interactive wizards. This paper describes an architecture for Internet-based education delivery systems that provide asynchronous learner support and facilitated interactions. This architecture makes use of a distributed and mobile, multi-agent system, using artificial intelligence techniques. (Author)

Reproductions supplied by EDRS are the best that can be made
from the original document.

PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY

G.H. Marks

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

2B

An Agent Architecture for Asynchronous Learning

Stephen Rochefort
Department of Electrical and Computer Engineering
University of Calgary
Calgary, AB T2N 1N4
rochefor@enel.ucalgary.ca

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

Minor changes have been made to improve reproduction quality.

Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Abstract: Education delivery systems have evolved from content delivery mechanisms to collaborative environments. The next step in this evolution is to provide greater support for asynchronous learners, those that progress through course material with little or no support from the instructional team. The integration of web technologies with artificial intelligence techniques and multi-agent environments provides an architecture that supports the development of asynchronous-based systems. Autonomous agents are capable of providing ease of use and guidance through the introduction of natural language interfaces, knowledge processing, inferential reasoning and interactive wizards.

ED 479 604

Introduction

Research and the application of distance education techniques has been the center of much attention in recent years. Numerous educational institutes are now providing some form of distance education curriculum. For example, Athabasca University in St. Albert, Alberta uses networking software to offer correspondence degrees. Memorial University in Newfoundland offers courses using audio-conferencing with computer linkups. Video-conferencing is also widely used since Queens's University in Kingston, Ontario, pioneered its use for a master of business administration program.

Distance education technologies have progressed from the user of paper materials to audio and video conferencing to Internet delivery, with many variations in between. The latest technology, the Internet, provides up-to-date materials retrievable instantly on-line from sources across the globe.

Advancements in the use of the Internet for distance education have focused on the publishing environment that the World Wide Web has become. Instructional teams are able to make course materials available on-line. More recent advancements have delved into supporting advanced pedagogies based on the principles of active learning, collaboration, multiplicity, and knowledge building (Harasim, 1995; Harasim, 1996). These educational systems facilitate course design, management, and evaluation, by providing software to enter and maintain, for instance, grade information or shared documents. They also provide virtual conferencing facilities, which allows for messages to be broadcast to virtual groups, which then can be organized by reply threads in order to follow up on a given discussion subject. Paramount behind these concepts is the idea of engaging learners in cooperative rather than competitive pursuit of knowledge.

Unfortunately, education delivery through the Internet stills falls short of the great possibilities that exist. There are two essential extensions that are required to harness the medium. First, current systems have little or no support for asynchronous learners who require support that enables them to find their own solutions and approaches to learning the course material. Second, communication in distance delivery systems still proceeds on an individual-to-computer basis with the learning software in charge of organizing the communications into coherent threads that the users can consult individually. Simultaneous interaction between various students and teachers is simulated and mediated, rather than facilitated, by the software.

This paper describes an architecture for Internet-based education delivery systems that provide asynchronous learner support and facilitated interactions. This architecture makes use of a distributed and mobile, multi-agent system, using artificial intelligence techniques. The architecture has been implemented and tested in the Logic Programming in Education for Asynchronous Learning Environments (LPed ALE) system.

IR022208

BEST COPY AVAILABLE

LPed ALE

The Logic Programming in Education for Asynchronous Learning Environment (LPed ALE) system is a full distance education delivery tool (Rochefort & Dahl, 1999; Rochefort et al, 2000). It provides standard education delivery facilities such as content delivery, communication, and course management and evaluation. Beyond these basic features, LPed ALE provides tools that support asynchronous learning through real time communication, a natural language interface allowing the user to control the tools with natural language, concept based searching and indexing, and interactive software agents that work with the user in their activities in the system.

LPed ALE expands previous work on the LogiMOO virtual world for distributed group-work over the Internet (Rochefort et al, 1998). These systems integrate web technology, via the Java programming language, and artificial intelligence techniques, via the Jinni logic programming language (Tarau, 1999). The integration allows for greater support of natural language interfaces, knowledge processing and interactive student assistant wizards. The use of artificial intelligence techniques allows for the development of systems that provide a more interactive and supportive environment for asynchronous learners. Furthermore, these techniques allow knowledge about the interactions to be captured and used to guide the learner through their experiences within the system.

The natural language interface uses a controlled subset of English that is translated into commands that are executed in the environment (Rochefort et al, 1997). Natural language processing techniques such as anaphora, relatives and multi-sentential input through an original mechanism, Assumption Grammars (Dahl et al, 1996; Tarau et al, 1996), enable the system to keep contextual information about the user's interactions and customize responses and queries to the user.

Concept-based hierarchies are used to index source and, in turn, allow the user to query the system for information. When a user queries the system through the natural language interface, concepts of interest are extracted from the semantics of the text and then matched to concepts listed in the index hierarchies. Sources that match a user's interest and intent in the query are then presented. Concepts can further be generalized or specialized to assist in locating relevant information.

Interactive student wizards are provided through agent technologies. These mobile software robots provide asynchronous support by traveling to student computers to interact with the student, receiving queries and determining appropriate responses, and interacting with the student to solve their problem(s). The agents are capable of traveling to course machines in order to process knowledge bases and to locate information and answers to student queries.

Supporting Asynchronous Learners

The basic components of an educational environment include ease of communication, ease of presentation, and ability to expand. For a cooperative learning environment, these basic components are extended with multi-user capabilities and interactivity. For asynchronous learners, there must be support for guiding students through a learning path with little or no interaction from the instructional team.

The abilities to organize shared information intelligently and to provide guided and response system support are paramount for the feasibility of such objectives. Systems that do not possess any kind of inferential capability ultimately fail the task. Logic programming, or other automated deduction mechanisms, will come to be recognized as the necessary way to extend educational delivery systems.

LPed ALE provides the basic components of educational environments through the web-based facilities that support extensible material presentation. It also supports collaborative work, allowing students working on the same project to interact amongst themselves and the instructional team. Asynchronous support is offered in LPed ALE through a system of collaborating and mobile agents.

LPed ALE implements asynchronous support through various features. The natural language interaction hides the complexities involved in interacting with a large software system and it encourages conversation interactions which can later be extended to voice-based interactions. A frequently asked questions (FAQ) knowledge base allows

students to ask and receive answers in an automated way. The student assistant agents is a mobile software robot that interact with students to lead them down a path of learning and uses mobility and collaborative techniques to assist the student. An instructional assistant agent is used to administer instructional goals, such as interactive examinations, to a group of students.

LPed ALE Architecture

The LPed ALE system is a web-based, collaborative multi-agent environment implemented in Jinni (Tarau, 1999), a lightweight, multi-threaded, logic programming language, intended to be used as a flexible scripting tool for gluing together knowledge processing components and Java objects in distributed applications. Jinni allows the web-based components and the artificial intelligence components to be isolated. The result is that static aspects of the system such as the user interface are created only once in Java while the dynamic aspects of the system such as the agents, which are implemented using artificial intelligence techniques, can be updated while the system is executing.

Multi-Agent Systems

Agent and Multi-Agent Systems (MAS) are a new software development paradigm where the principle idea is that software systems can be broken down into small, efficient, and autonomous software robots. These agents work on behalf of a user to solve more complex computing problems than those achieved by more conventional algorithmic programming. An inherent characteristic of MASs is the spatial and temporal distribution of processing knowledge.

Agent systems have advanced from the basic notions of beliefs, goals, plans, roles, and groups. The intent is that agents are "individuals" that have goals and responsibilities that are accomplished in an autonomous nature. In order to accomplish goals, agents make use of knowledge about themselves, other agents and the environment in which they exist in order to develop beliefs, plans, and roles.

Distance education delivery systems can benefit from multi-agents systems because of their inherent nature of distributed processing and knowledge. The use of agents creates a partnership with the learner and together the two can address learning issues encountered in asynchronous learning. In order to take full advantage of multi-agent systems, education delivery systems require an architecture that supports the various aspects of asynchronous learning.

Education Delivery System Architecture

The LPed ALE multi-agent system architecture (Figure 1) consists of three main components. A Java manager application executes on course servers to provide services to the instructional team including content delivery, student management, and agent configuration. On each student machine, a Java manager application provides a portal for the student to retrieve course materials, complete course assignments and interact with system agents. The third component consists of the mobile, collaborative agents that provide asynchronous learner support.

LPed ALE Manager Applications

The *Site Manager* is a Jinni process managing a blackboard of information related to all course sites and student sites that exist. In either case, when a course site becomes active to offer its contents or when a student comes on-line, these two types of sites must register with the Site Manager. This effectively makes the Site Manager a global "phone book" of all course sites and student sites currently active. Once a site has registered with the Site Manager, it can participate in all networked activities. This includes information access, mobile agent interactions, and chat.

The Site Manager must be located on a computer that makes accessing optimal. This could be a remote machine unrelated to a specific course site or student site but typically it would be the main site where an institution would set up its courses.

The *Student Manager* is a Java process responsible for providing access to core activities that a student would require to use the system. This process is located on a networked computer set up to interact with web-based delivery sites. A Student Database is kept on the same site to store all information collected during student activities (student notes, communications, agent interaction results) as well as maintaining a student profile. A student profile contains information specific to the student including login parameters, course enrollment identification, along with data on student activities.

The Student Manager provides a controlled natural language interface through a web-browser. This interface allows students to interact with the system using a natural language such as English. A natural language interface enhances the point and click nature of the interface. This provides students with the opportunity to interact and directly manipulate the environment.

The activities that can be performed through the Student Manager include:

- Course registration/deregistration/login/logout.
- Course participation (accessing and participating in course activities).
- Support agent access.
- Chat.

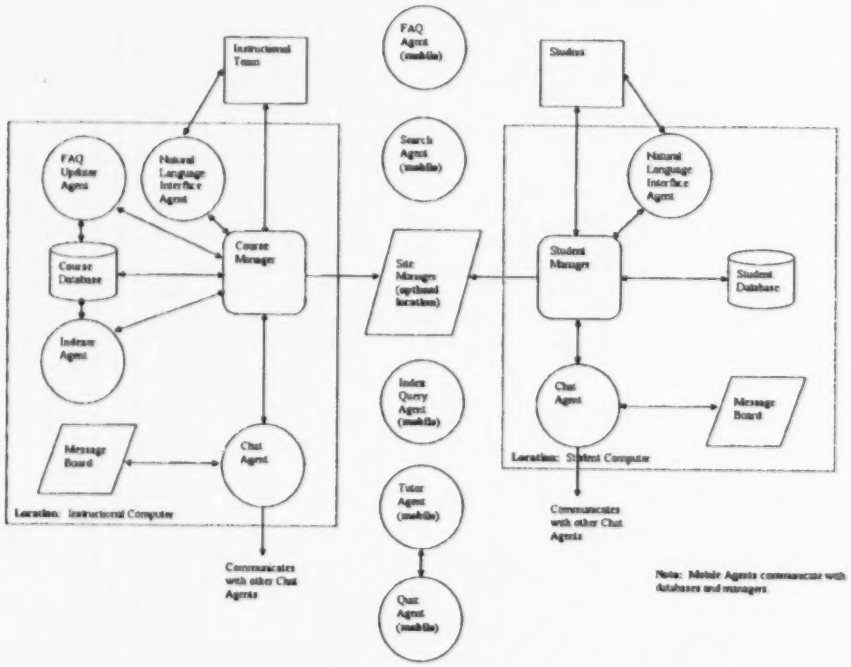

Figure 1: LPed ALE Multi-Agent System Architecture

The *Course Manager* is a Java process responsible for providing access to the core activities that an instructional team would be involved in to set up and administer a course site. This process is located on a networked computer set up to deliver web-based material. A Course Database is kept on the same site to store all information about the course(s) (lecture notes, demos, frequently asked questions, instructional team profiles, and any links to information resources) as well as maintain information related to course statistics, and student grades and profile information.

Each course on the course site is given a course profile which provides links to the different components of the course information set. This profile is also used to store statistical information such as timestamps indicating the last update and page access information.

BEST COPY AVAILABLE

The Course Manager includes a controlled natural language interface that allows the instructional team to interact with the support agents (FAQ, search, etc.) just as a student would. This is useful for testing facilities from a student's point of view.

The activities that can be performed through the Course Manager include:

- Course creation and management.
- Course enrollment.
- Grade management.
- New course material indexing.
- Frequently Asked Questions knowledge base update.
- Support agent access.
- Chat.

LPed ALE Agents

As described in earlier, agents are modules of software, situated within a computing system, that are knowledgeable of the environment and their responsibilities within that environment, and are capable of autonomous actions to accomplish goals. Within LPed ALE there are numerous agents, implemented in Jinni, interacting with the external entities, databases, and managers. The agents have a key role in providing assistive support to all external entities (i.e. the instructional team and students).

The *Natural Language Interface (NLI) Agent* is responsible for tokenizing and parsing controlled natural language input, in the form of English statements, into system commands and then invoking those commands. This agent exists in both the course application alongside the Course Manager and in the student application alongside the Student Manager.

The NLI Agent communicates with the respective manager, which maintains a kernel of commands, to invoke system commands. These invocations can then lead to information being retrieved and displayed or agents being activated or deactivated.

The *Chat Agent* is responsible for processing incoming and outgoing chat messages. Incoming messages received from other Chat Agents are placed on the *Message Board* until they are processed. When they are processed, the Chat Agent presents the incoming messages to the user through the web browser interface. These messages are formatted to show the sender of the message.

Outgoing messages from the User are stored on the Message Board until they are processed. At processing, the Chat Agent looks up the recipient of the message through the Site Manager and then the message is passed to the respective Chat Agent at the recipient's site.

The *Message Board* is a process managing a local blackboard of chat messages. This blackboard stores the messages being sent from and sent to the local user. These messages are managed by the Chat Agent. All messages are stored with sender and receiver information to make identification easy.
The *Frequently Asked Questions (FAQ) Agent* is a mobile agent that processes queries on a FAQ knowledge base. A user activates the FAQ agent through the respective manager's NLI Agent by asking a question in a controlled form of English. The NLI Agent will parse the question and log it with the FAQ Agent through a system command in the respective manager.

When a FAQ Agent receives a query, it parses the query and then searches for related questions in the FAQ knowledge base. When a match is found, corresponding answers are retrieved and returned to the respective manager which then displays the results.

The FAQ knowledge base stores representations of questions in categories based on their relevant topic. The relevant topic is the key to making a match between the user's question and the question-answer pairs stored in the knowledge base. The result returned to a user will be all question-answer pairs that correspond to the topic of the user's question.

The *Search Agent* is a mobile agent that receives queries via the NLI Agent and the respective manager. When a query is logged with the Search Agent, the agent will return to the Course Manager to compare the search parameters with the categorization of course documents stored in the course database. The resulting matches will be returned to the respective manager to present the user with a list of links that correspond to related documents. These links are presented in the web browser portion of the user interface thus allowing the user to click on the link and load the respective document.

The *Index Query Agent* is a mobile agent that is used to provide concept hierarchy information to requesting users. The Course Database contains a concept hierarchy related to the specific course. This hierarchy ranges upwards in a computing science tree fashion with generalized concepts while ranging downwards with specialized concepts.

Users can make controlled English requests for a generalization or specialization of a specific topic. In the case of generalization the Index Query Agent returns the parent of the specified topic. In the case of specialization the Index Query Agent returns a list of child concepts.

The *Tutor Agent* is a mobile agent responsible for interacting with a user through a question-answer paradigm. The user is able to make queries of the Tutor Agent via the NLI agent and the respective manager. When the Tutor Agent receives a question, it parses it and determines a course of action and where possible solutions may exist. The agent then moves to and interacts with the Course Manager and other agents to resolve the requests. The interaction between user and Tutor Agent continues until the user asks the Tutor Agent to deactivate.

The Tutor Agent acts as a facilitator of the other agents. Dependent on the query, the Tutor Agent may communicate with one or more of the other support agents in order to provide answers and further interaction.

The *Quiz Agent* is a mobile agent responsible for delivering quizzes to students. The quizzes are configured by the instructional team. A set of questions can be entered by the instructional team. Then the team constructs a test by assigning a set of questions to a test. Finally, the team can identify which students are to take which test.

Once the quiz agent has been configured, it waits for specified students to log in. When a student logs in and they are required to take a quiz, the Quiz Agent activates and starts a quiz.

When a student is taking a quiz the agent receives answers from the student and compares their answer to the correct answer. If the answers match, the agent continues with the next question or ends if there are no more questions. If the answers do not match, the quiz agent communicates the topic of the question to the Tutor Agent that then activates and starts an interaction with the student. When the student is done interacting with the Tutor Agent, and hopefully has a better understanding of the particular topic, the Quiz Agent resumes with the next question.

Status of the LPed ALE System

The LPed ALE system is a fully operational system with an implementation of the aforementioned architecture and agents integrated into the core distance delivery system. The system has been tested on a small subset of students from an Introduction to Computer Science class. Preliminary results indicate that students have a preference to the LPed ALE system over static delivery systems because some of their basic knowledge discovery questions and concerns could be handled by the system using the natural language interface and the supporting agent infrastructure. The results also indicate a need for a more elaborate support mechanism including handling exploratory questions in topical areas related to the core course materials, providing flexibility in the exploration scenarios, and providing a more robust set of semantics in the natural language interface.

Conclusions

The LPed ALE system is an imp lementation of a multi-agent system architecture. With the advent of autonomous agents, educational delivery systems will see a greater support for asynchronous learners. These agents will interact with and guide students as they progress through the learning process. They will act as surrogates for the instructional team providing answers, advice, and direction.

The future of systems such as LPed ALE will involve the exploration of agent usability. If a platform can be provided that enables end-users, such as the instructional team, to program the agents without the need for highly specialized training we could expect a fundamental change in the use of the educational delivery systems. If we consider the potential of a large population of instructional members and institutes capable of programming agents in these systems, we can expect a significant increase in the quality and the use of such systems..

It is through the development of easy to use multi-agent systems that educational environments can be expanded to support the natural learning behaviors of distance students. The use of artificial intelligence techniques is key to extended educational systems because of the intrinsic knowledge processing nature. Since course delivery is the passing of large amounts of knowledge it only makes sense to use tools that can manipulate the knowledge effectively.

References

[1] Dahl, V., Fall, A., Rochefort, S., and Tarau, P. (1996). *A Hypothetical Reasoning-Based Framework for NL Processing*. 8th International Conference on Tools with Artificial Intelligence.

[2] Harasim, L., Calvert, T., and Collings, D. (1996). *A Networked Learning Environment for Continuing Engineering Education*, The Professional Engineer, 47.

[3] Harasim, L. (1995). *The Virtual University: New Approaches to Higher Education in the 21st Century*, Australian Society for Computers in Learning in Tertiary Education Conference.

[4] Rochefort, S. and Dahl, V. (1999). *LPed ALE: An Application of Logic Programming in Distance Education*. International Conference on Logic Programming, Workshop on Distributed and Internet Programming with Logic and Constraint Languages, Las Cruces, New Mexico.

[5] Rochefort, S., Dahl, V., and Tarau, P. (1998). *A Virtual Environment for Collaborative Learning*. Systematics, Cybernetics, and Informatics, Orlando, Florida.

[6] Rochefort, S., Dahl, V., and Tarau, P. (1997). *Controlling Virtual Worlds Through Extensible Natural Language*. AAAI Symposium on Natural Language Processing for the World Wide Web, Palo Alto, California.

[7] Rochefort, S., Dahl, V., and Tarau, P. (2000). LPed *ALE: New Trends in Asynchronous Learning*. 6th International Conference on Information Systems Analysis and Synthesis, Orlando, Florida.

[8] Tarau, P. (1999). *Prolog Interpreter in Java: Jinni – Online Documentation*. Binnet Corporation. Note: http://www.binnet.com/Jinni.

[9] Tarau, P., Dahl, V., and Fall, A. (1996). *Backtrackable State With Linear Affine Implication and Assumption Grammars*, Proceedings of Asian Computing Science Conference.

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

Reproduction Basis

[X] This document is covered by a signed "Reproduction Release (Blanket)" form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.

[ ] This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").